

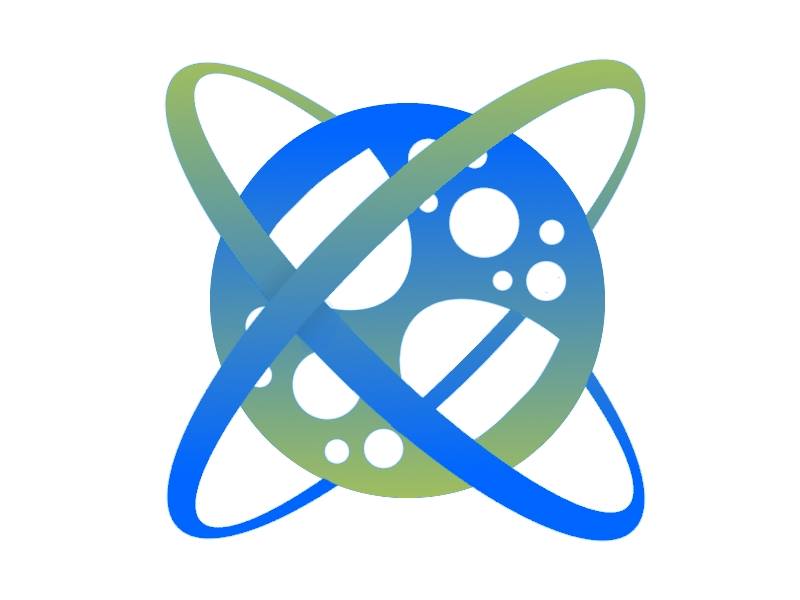
**Software Process, SOEN 341/4 S, Winter 2016**

Dr. Shang

Dr. Fancott

Mr. Morse

**TimeTurner** by team YAWD

****

**Project Design Document – Deliverable 2**

|  |  |
| --- | --- |
| **Team members information** | |
| **Name** | **SID** |
| **Daniel Di Corpo** | **26331602** |
| **Dimitri Topaloglou** | **29358269** |
| **Claudia Della Serra** | **26766048** |
| **Philip Lim** | **27485506** |
| **Aline Koftikian** | **27764162** |
| **Ryan Lee** | **27752504** |
| **Erin Benderoff** | **27768478** |
| **Ideawin-Bunthy Koun** | **26314155** |
| **Kevin Yasmine** | **27195346** |
| **Marc-Andre Leclair** | **27754876** |
| **Lori Dalkin** | **27738293** |
| **Bryce Drewery-Schoeler** | **27283199** |

Grading Sheet

|  |  |  |
| --- | --- | --- |
| Section | Evaluation criteria (see instructions in the template for details) | Grading |
| all | 10 marks are allocated for excellence, professionalism and quality of work above and beyond the correct meeting of specifications.. | /10 |
| 1 | Presentation of this document | /5 |
| 2 | Completeness and accuracy with regard to initial project description | /1 |
| 3.1 .  .  3.2  3.3 | Completeness and accuracy of the project functional requirements expressed as formal use cases, including difficulty and importance indicators  completeness and accuracy of the diagram and description of the domain model  completeness and accuracy with regard to initial project description accuracy with regard to initial project description, difficulty and importance ratings | /15  .  /3 .  /1 |
| 4.1 | Description of all team members’ capacities and schedule restrictions | /1 |
| 5 | List of goals removed from the project.  For each goal removed, give justifications in light of the resources available | /`1 |
| 6.1 .  6.2 | Clarity of textual description, validity of rationale, clarity and appropriateness of diagram, list of modules responsibilities  List of technologies used, validity of rationale | /2 .  /1 |
| 7.1 .  7.2 .  7.3 .  7.4  7.5  7.6 | Completeness of list of activities, clarity of their stated purpose, as well as statement of what artifacts they are producing  Completeness of list of artifacts to be produced during the project, validity of roles description of each artifact  Cost estimation of each individual artifact, validity of explanation of cost estimation, total cost estimate  Mapping of activities to individual project members  Accurate and complete presentation of milestones  *Assessment of risks `* | /1 .  /2 .  /2 .  /1  /1  /1 |
| 8 | Early Prototyping | /2 |
| Total |  | /50 |

DO NOT REMOVE THIS PAGE WHEN SUBMITTING YOUR DOCUMENT

**TABLE OF CONTENTS**

[1. Introduction 4](#_Toc446021738)

[2. Project Description 4](#_Toc446021739)

[3. Architectural Design 4](#_Toc446021740)

[3.1 Architecture Diagram (4+1 View) 4](#_Toc446021741)

[3.1.1 Physical View 4](#_Toc446021742)

[3.1.2 Logical View 5](#_Toc446021743)

[3.1.3 Process View 5](#_Toc446021744)

[3.1.4 Development View 5](#_Toc446021745)

[3.1.5 Situational View (Use Cases) 5](#_Toc446021746)

[3.2 Subsystem Interfaces Specifications 6](#_Toc446021747)

[4. Detailed Design 6](#_Toc446021748)

[4.1 Detailed Design Diagram 6](#_Toc446021749)

[4.2 Unit Descriptions 6](#_Toc446021750)

[5. Dynamic Design Scenarios 6](#_Toc446021751)

[6. Estimation 6](#_Toc446021752)

[7. Rapid Prototyping and Risk 6](#_Toc446021753)

# Introduction

# Project Description

The proposed and outlined web application, known as TimeTurner, is designed to auto-generate a student’s course sequence from their first semester up until the end of their degree. It takes into account user input preferences and and any previously completed courses or course prerequisites before creating this sequence. Preferences can be made by the student and include options such as night classes or having particular days off. The application will notify the user if a certain preference suggested results in an impossibility or conflict in the sequence. This sequence generator will be able to create a sequence at any point throughout the user’s degree, if sudden change in circumstances were to arise.

The goal of this application is to simplify the method with which students may decide and schedule their courses. If a course must be redone, the generator can decide what other courses should move where in regards to the remaining courses to be completed, which can be done in seconds, rather than hours. It saves the time of the user, in a simple and efficient manner. Ultimately, the system’s end goal will be to simplify a student’s task of creating their own course schedule in order to allow students to redirect their time to other more important activities, thus making course registration much simpler, quicker, and easier.

# Architectural Design

## Architecture Diagram (4+1 View)

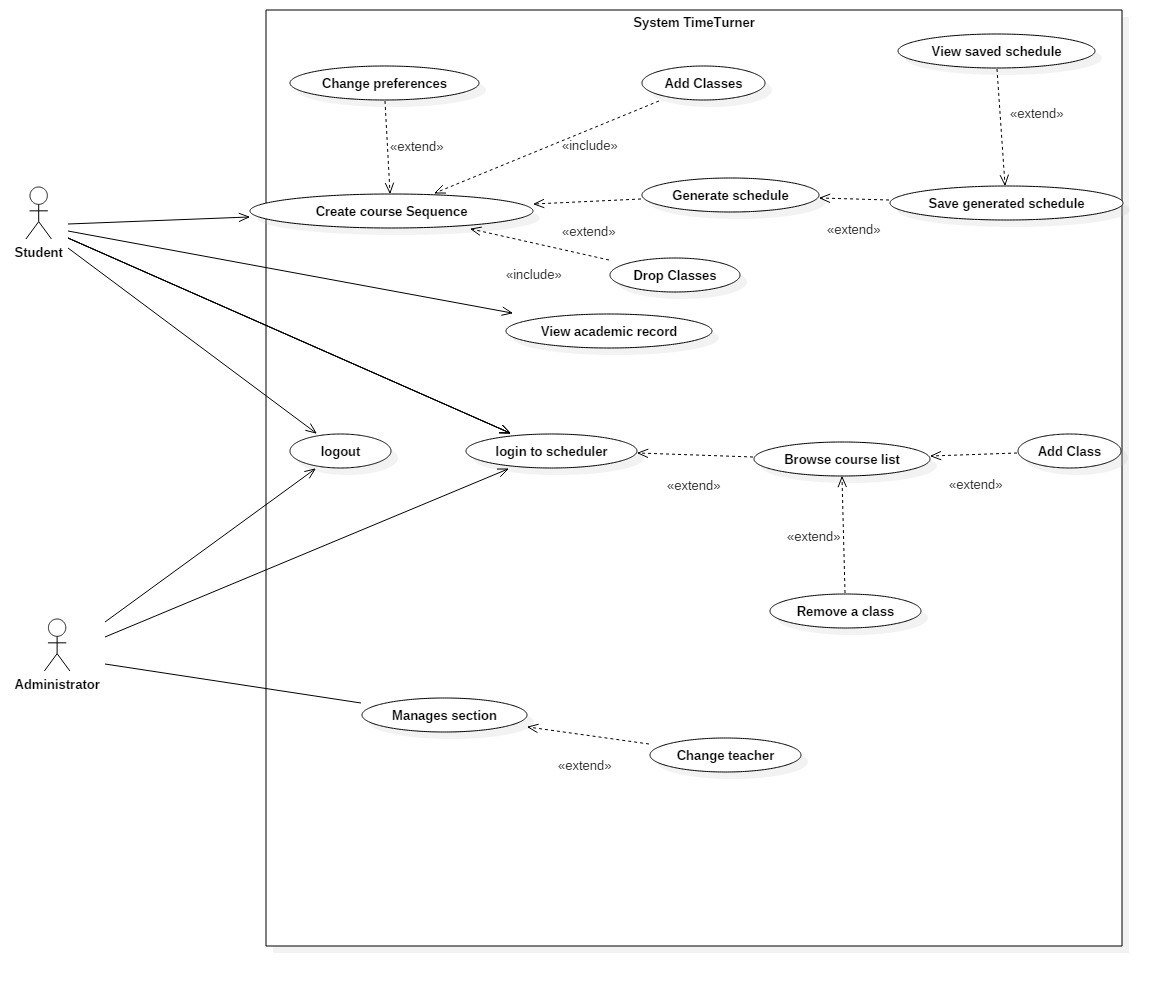
Physical View

Logical View

Process View

Development View

Situational View (Use Cases)



## Subsystem Interfaces Specifications

# Detailed Design

## Detailed Design Diagram

## Unit Descriptions

# Dynamic Design Scenarios

5.1 Use Case 1: Login to Time Turner

5.1.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC1 | |
| Use Case Name: | Login to TimeTurner | |
| Created By: | Marc-Andre Leclair | Last Updated By: Claudia Della Serra |
| Date Created: | January 25th, 2016 | Last Revision Date: February 8, 2016 |
| Actor(s): | Student, Administrator | |
| Goal/Actor Goals: | A student or Administrator wants to Login in their account | |
| Description/Summary: | The Student or Administrator wants to login into their account. The initial webpage contains a login box where the he or she can enter their username and password. The request is sent and it is check in the database whether or not the account exist or if the password is correct | |
| Preconditions: | * The user has an internet connection. * The user has valid login credentials. | |
| Post-conditions: | The user is authenticated and logged in. | |
| Minimum Guarantee: | User will not log in | |
| Basic Flow: | 1. Student enters its login information 2. Server verifies if it is in the database 3. Student is logged in and brought to the home page. | |
| Risk assessment: | Low | |
| Importance assessment: | 5/5 | |

5.1.2 System Sequence Diagrams

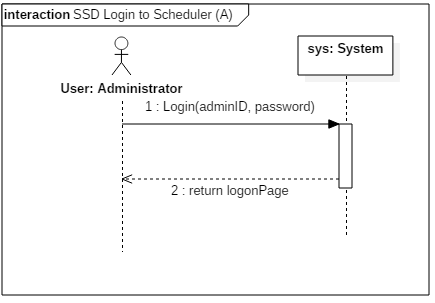


Figure 5.1.1 Administrator login system sequence diagram

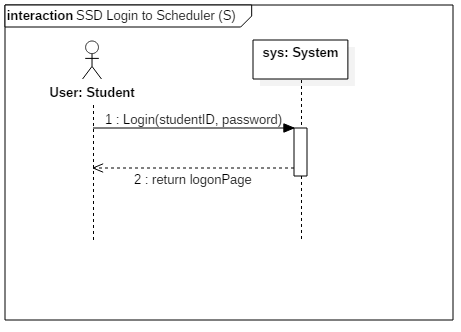


Figure 5.1.2 Student login system sequence diagram

5.1.3 Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 1.1 | Login | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | login(adminID: long) | |
| Cross-reference: | UC1 | |
| Preconditions: | * The user has an internet connection. * The user has valid login credentials. | |
| Post-conditions: | * The user is logged into their account. (instance creation) * A connection to the servers is established. (association creation) | |

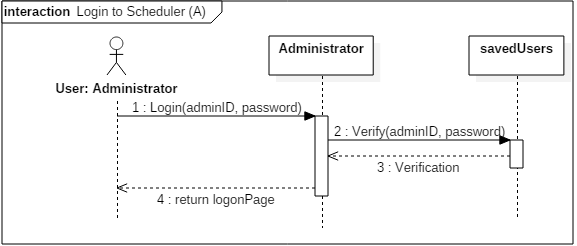


Figure 5.1.3 Administrator login contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 1.2 | Login | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | login(studentID: long) | |
| Cross-reference: | UC1 | |
| Preconditions: | * The user has an internet connection. * The user has valid login credentials. | |
| Post-conditions: | * The user is logged in. (instance creation) * A connection to the servers is established. (association creation) | |

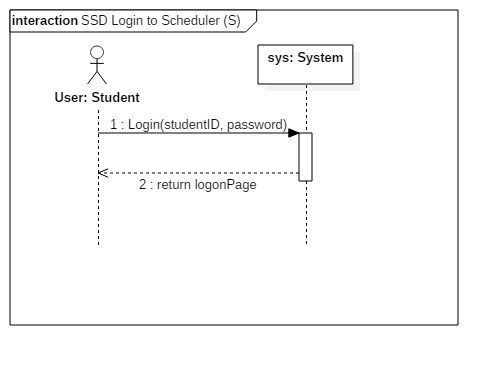


Figure 5.1.4 Student login contract diagram

5.2 Use Case 2 : Logout of TimeTurner

5.2.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC2 | |
| Use Case Name: | Logout of Scheduler | |
| Created By: | Marc-Andre Leclair | Last Updated By: Ryan Lee |
| Date Created: | January 25th, 2016 | Last Revision Date: March 17, 2016 |
| Actor(s): | Student, Administrator | |
| Goal/Actor Goals: | A student or Administrator wants to Logout of their account. | |
| Description/Summary: | The Student or Administrator wants to logout of their account. He or she sends a request to the server to close the connection between themselves and the Time Turner. | |
| Preconditions: | The user is logged in their account | |
| Post-conditions: | The user is disconnected from the server and logged out. | |
| Minimum Guarantee: | User will remain logged in | |
| Basic Flow: | 1. The user indicates that they wish to logout of the system. 2. The server terminates the connection. 3. The user is logged out through the webpage. | |
| Risk assessment: | Low | |
| Importance assessment: | 5/5 | |

5.2.2 System Sequence Diagrams

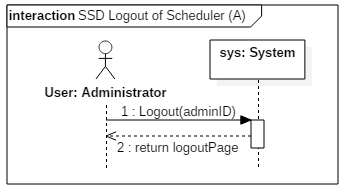


Figure 5.2.1 Administrator logout system sequence diagram

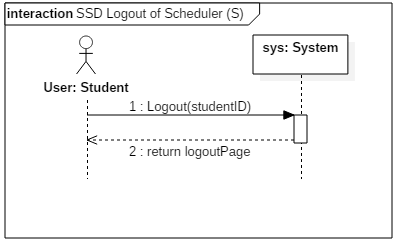


Figure 5.2.2 Student logout system sequence diagram

5.2.3 Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 2.1 | Logout | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | logout(adminID: long) | |
| Cross-reference: | UC2 | |
| Preconditions: | * The user is logged into their account. | |
| Post-conditions: | * The user is disconnected from the server and logged out. (instance deletion) | |

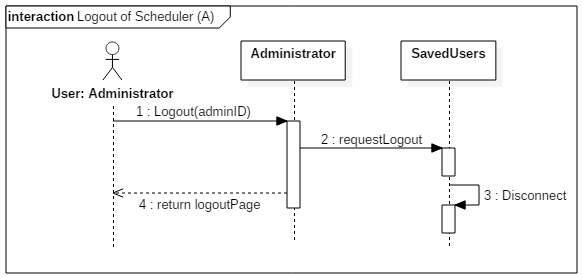


Figure 5.2.3 Administrator logout contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 2.2 | Logout | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | logout(studentID: long) | |
| Cross-reference: | UC2 | |
| Preconditions: | * The user is logged into their account. | |
| Post-conditions: | * The user is disconnected from the server and logged out. (instance deletion) | |

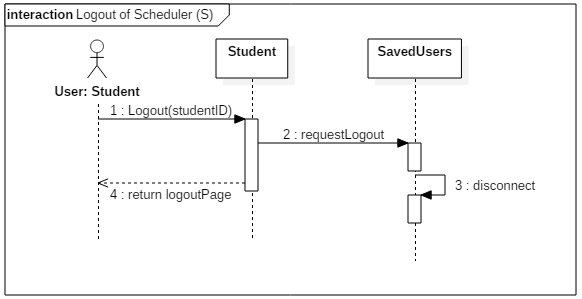


Figure 5.2.4 Student logout contract diagram

5.3 Use Case 3: Create Course Sequence

5.3.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC3 | |
| Use Case Name: | Create Course Sequence | |
| Created By: | Marc-Andre Leclair | Last Updated By: Ryan Lee |
| Date Created: | January 25th, 2016 | Last Revision Date: March 9, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | A student wants to create a course sequence | |
| Description/Summary: | The Student wants to create his or her course sequence. The system must provide the user with the option to add their own preferences in order to generate a sequence that is personalized to these preferences. | |
| Preconditions: | * The user is logged into the system * The user is registered in a program. | |
| Post-conditions: | The user has a personalized course sequence for the future | |
| Minimum Guarantee: | No course sequence will be created | |
| Basic Flow: | 1. Student requests to create a sequence. 2. The system must prompt the student to add his or her preferences to the sequence. 3. The student selects zero, one, or more preferences. 4. The system prompts the student to confirm their preferences 5. The student confirms his choices. 6. The course sequence is created and shown to the student. | |
| Risk assessment: | High | |
| Importance assessment: | 5/5 | |

5.3.2 System Sequence Diagram

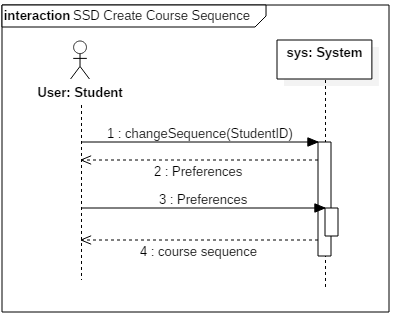


Figure 5.3.1 Create course system sequence diagram

5.3.3 Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 3.1 | getPreference | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | getPreferences( ) | |
| Cross-reference: | UC3 | |
| Preconditions: | * The user is logged into the system * The user is registered in a program. | |
| Post-conditions: | none | |

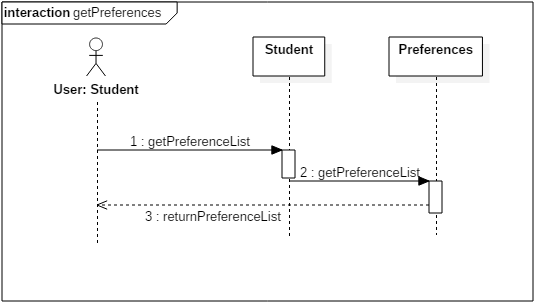
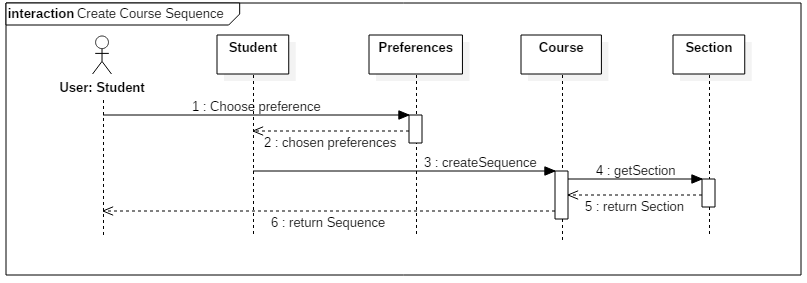


Figure 5.3.2 getPreferences contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 3.2 | createCourseSequence | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | createCourseSequence(studentID: long) | |
| Cross-reference: | UC3 | |
| Preconditions: | * The user is logged into the system * The user is registered in a program. | |
| Post-conditions: | * The user has a personal sequence created for them. (attribute modification, association creation) | |



5.4 Use Case 4: Browse Course List

5.4.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC4 | |
| Use Case Name: | Browse Course List | |
| Created By: | Ideawin-Bunthy Koun | Last Updated By: Ryan Lee |
| Date Created: | January 25th, 2016 | Last Revision Date: March 9, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | Browse the list of courses from the course calendar. | |
| Description/Summary: | The user wishes to view available courses and access information pertaining to them. The system must display such information, including sections, times, and locations. | |
| Preconditions: | * User must be logged in as a Student. * Course list must be available. | |
| Post-conditions: | A list of courses is displayed. | |
| Minimum Guarantee: | The system fails to display a list of courses and displays an error message to the user. | |
| Basic Flow: | 1. Student selects the ‘Browse Course List” feature. 2. System retrieves list of courses. 3. System displays list of courses. 4. User selects a course from that list. 5. System displays information about the selected course. | |
| Risk assessment: | Medium | |
| Importance assessment: | 3/5 | |

5.4.2 System Sequence Diagram

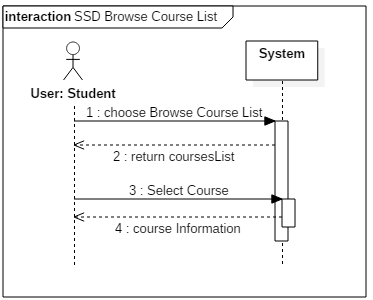


Figure 5.4.1 Browse course list system sequence diagram

5.4.3 Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 4.1 | getCourseList | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | getCourseList( ) | |
| Cross-reference: | UC4 | |
| Preconditions: | * User must be logged in as a Student. * Course list must be available. | |
| Post-conditions: | None | |

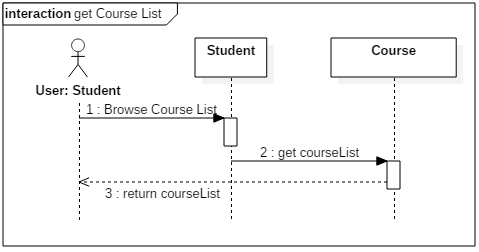


Figure 5.4.2 getCourseList contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 4.2 | getCourseInfo | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | getCourseInfo(courseID: long) | |
| Cross-reference: | UC4 | |
| Preconditions: | * User must be logged in as a Student. * Course list must be available. | |
| Post-conditions: | None | |

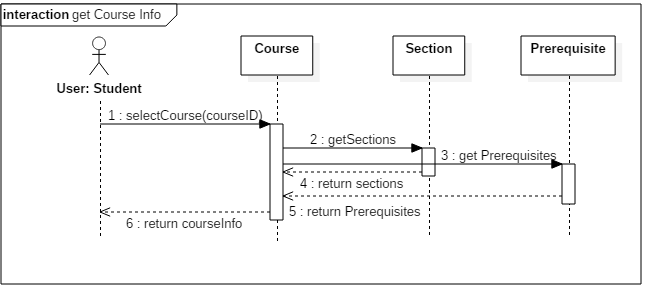


Figure 5.4.3 getCourseInfo contract diagram

5.5 Use Case 5: View Course Sequence

5.5.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC5 | |
| Use Case Name: | View Course Sequence | |
| Created By: | Ideawin-Bunthy Koun | Last Updated By: Ryan Lee |
| Date Created: | January 25th, 2016 | Last Revision Date: March 9, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | Allows the user to view their expected progression in the program. | |
| Description/Summary: | User can view a complete course sequence of the program the student is enrolled in. The Sequence will show the courses the Student is expected to take, and the order he or she is expected to take them in. | |
| Preconditions: | * User must be logged in as a Student * The system has access to a course sequence. | |
| Post-conditions: | The system displays a course sequence. | |
| Minimum Guarantee: | The system fails to display a course sequence and displays an error message to the user. | |
| Basic Flow: | 1. Student selects the option to view his or her course sequence. 2. System retrieves the sequence. 3. Course sequence is displayed to the student. | |
| Risk assessment: | Medium | |
| Importance assessment: | 4/5 | |

5.5.2 System Sequence Diagram

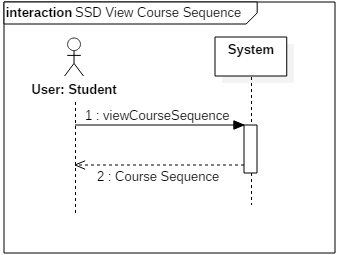


Figure 5.5.1 View Course Sequence system sequence diagram

5.5.3 Contract diagrams

|  |  |  |
| --- | --- | --- |
| Contract 5.1 | viewCourseSequence | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | viewCourseSequence(studentID: long) | |
| Cross-reference: | UC5 | |
| Preconditions: | * User must be logged in as a Student * The system has access to a course sequence. | |
| Post-conditions: | None | |

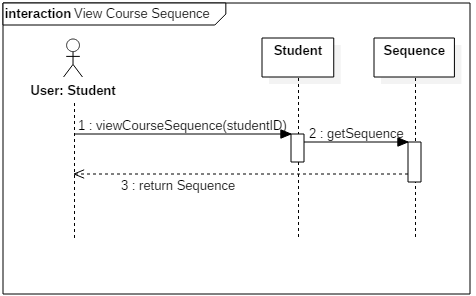


Figure 5.5.2 viewCourseSequence contract diagram

5.6 Use Case 6: Generate Schedule

5.6.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC6 | |
| Use Case Name: | Generate Schedule | |
| Created By: | Ideawin-Bunthy Koun | Last Updated By: Ryan Lee |
| Date Created: | January 25th, 2016 | Last Revision Date: March 9, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | Obtain a suggested schedule for the next four years. | |
| Description/Summary: | The system can generate a 4-year schedule for the user based on the student’s preferences and constraints and on prerequisite and co-requisite policy. | |
| Preconditions: | * User must be logged in as a Student. * The system must have access to course database. * The system must have access to the student’s records. | |
| Post-conditions: | The system generates a schedule with no overlaps. | |
| Minimum Guarantee: | The system fails to generate a schedule and displays an error message to the user. | |
| Basic Flow: | 1. User selects the ‘Generate Schedule’ feature. 2. System retrieves list of preferences. 3. User selects his or her options and preferences. 4. System prompts user to confirm preferences. 5. System responds by displaying a schedule that meets the user’s preferences and constraints. | |
| Risk assessment: | High | |
| Importance assessment: | 5/5 | |

5.6.2 System Sequence Diagram

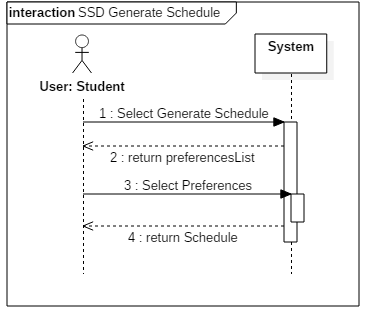


Figure 5.6.1 Generate Schedule System Sequence Diagram

5.6.3 Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 6.1 | getPreferences | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | getPreferenceList | |
| Cross-reference: | UC6 | |
| Preconditions: | * User must be logged in as a Student. * The system must have access to course database. * The system must have access to the student’s records. | |
| Post-conditions: | None | |

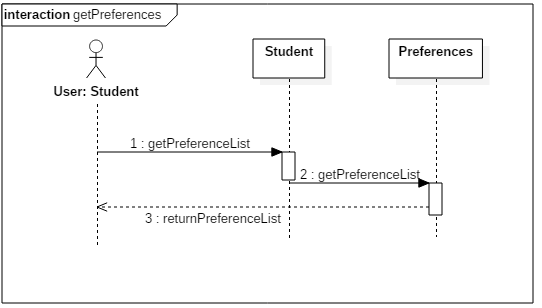
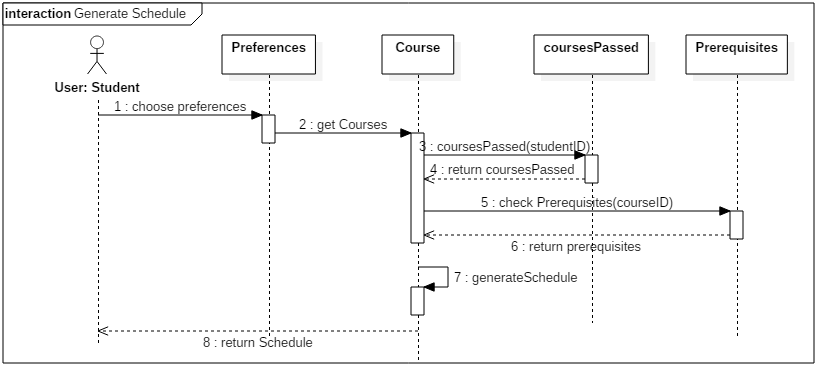


Figure 5.6.2 getPreferences contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 6.2 | generateSchedule | |
| Created By: | Ryan Lee | Last Updated By: Ryan Lee |
| Date Created: | March 17, 2016 | Last Revision Date: March 17, 2016 |
| Operation: | generateSchedule(studentID: long) | |
| Cross-reference: | UC6 | |
| Preconditions: | * User must be logged in as a Student. * The system must have access to course database. * The system must have access to the student’s records. | |
| Post-conditions: | * An instance of Schedule, schd, will be created (instance creation) * Schd will be added to student’s saved schedules. (attribute modification, association creation) | |



5.7 Use Case 7 – View Saved Schedules

5.7.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC7 | |
| Use Case Name: | View Saved Schedules | |
| Created By: | Erin Benderoff | Last Updated By: Claudia Della Serra |
| Date Created: | February 7, 2016 | Last Revision Date: February 14, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | The user wishes to view a previously saved schedule. | |
| Description/Summary: | The user selects a schedule from a list of their previously saved schedules. The system must display this schedule for the user to review. | |
| Preconditions: | * User is logged into the system. * User has previously generated one or more schedules * User has saved at least one generated schedule | |
| Post-conditions: | The system displays the saved schedule | |
| Minimum Guarantee: | Saved schedules remain in the system unchanged. | |
| Basic Flow: | 1. The user indicates their wish to view a saved schedule 2. The system prompts the user with a list of saved schedules 3. The user selects which semester’s schedule to view 4. The system displays the chosen schedule | |
| Risk assessment: | Medium | |
| Importance assessment: | 3/5 | |

5.7.2 System Sequence Diagram

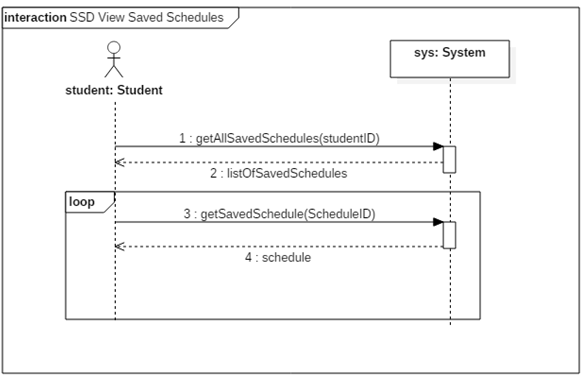


Figure 5.7.1 View Saved Schedules system sequence diagram

5.7.3 Operating Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 7.1 | getAllSavedSchedules | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | viewSavedSchedules(studentID: long) : list<SavedSchedule> | |
| Cross-reference: | UC7 | |
| Preconditions: | * User is logged into the system. * User has previously generated one or more schedules * User has saved at least one generated schedule | |
| Post-conditions: | None | |

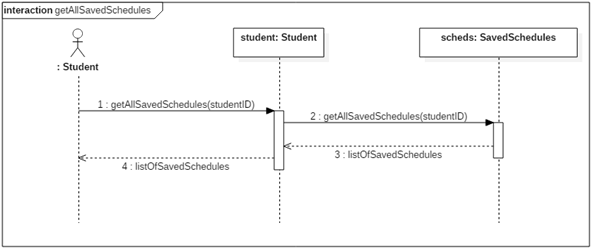


Figure 5.7.2 getAllSavedSchedules contract diagrams

|  |  |  |
| --- | --- | --- |
| Contract 7.2 | getSavedSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | getSavedSchedule(scheduleID: long) : Schedule | |
| Cross-reference: | UC7 | |
| Preconditions: | * User is logged into the system. * User has previously generated one or more schedules * User has saved at least one generated schedule | |
| Post-conditions: | None | |

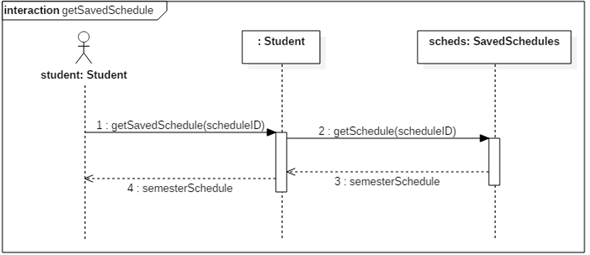


Figure 5.7.3 getSavedSchedule contract diagram

5.8 Use Case 8 – View Academic Record

5.8.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC8 | |
| Use Case Name: | View Academic Record | |
| Created By: | Erin Benderoff | Last Updated By: Claudia Della Serra |
| Date Created: | February 7, 2016 | Last Revision Date: February 14, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | The user wishes to view a list of his or her completed courses and grades for those courses. | |
| Description/Summary: | The system displays the user’s academic record, which includes completed courses and courses currently being taken. | |
| Preconditions: | * User is logged into the system. * User has completed at least one course and/or is presently enrolled in at least one course. | |
| Post-conditions: | The system displays the user’s academic record. | |
| Minimum Guarantee: | The academic record remains in the system unmodified. | |
| Basic Flow: | 1. The user requests to view their academic record 2. The system generates a list of the user’s completed courses, as well as courses in which the user is currently enrolled. | |
| Risk assessment: | Low | |
| Importance assessment: | 1/5 | |

5.8.2 System Sequence Diagram

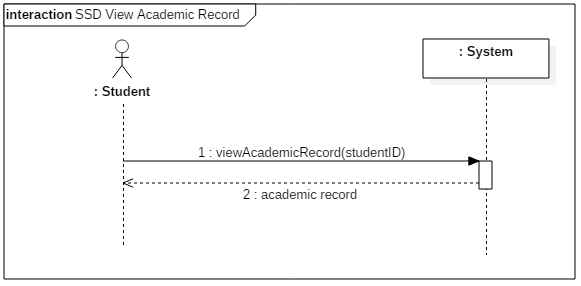


Figure 5.8.1 View Academic Record System Sequence Diagram

5.8.3 Operating Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 8.1 | viewAcademicRecord | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | viewAcademicRecord (studentID: long) : list<Course> | |
| Cross-reference: | UC8 | |
| Preconditions: | * User is logged into the system. * User has completed or is presently enrolled in at least one course | |
| Post-conditions: | * A list of classes passedis dynamically created for the student (instance creation) | |

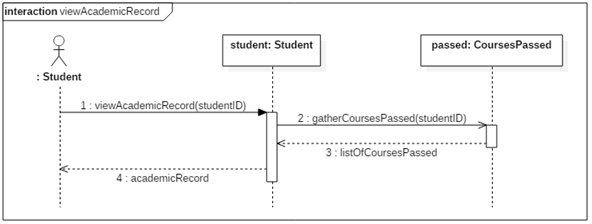


Figure 5.8.2 viewAcademicRecord contract diagram

5.9 Use Case 9 – Drop Course

5.9.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC9 | |
| Use Case Name: | Drop Course | |
| Created By: | Erin Benderoff | Last Updated By: Claudia Della Serra |
| Date Created: | February 7, 2016 | Last Revision Date: February 14, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | The user wishes to remove a course from a generated schedule. | |
| Description/Summary: | The user selects which course he or she would like to remove from the schedule. The system must delete this course from the schedule. | |
| Preconditions: | * User is logged into the system. * User has generated a schedule containing at least one course. * User has chosen to view a saved schedule | |
| Post-conditions: | The chosen course has been removed from the user’s schedule. | |
| Minimum Guarantee: | The courses on the user’s schedule remain unchanged. | |
| Basic Flow: | 1. User selects a schedule they wish to view 2. System returns the desired Schedule 3. The user chooses a course from his or her schedule to view. 4. The system displays the course information 5. The user indicates they wish to remove the course from the schedule 6. The System prompts the user for confirmation. 7. The user confirms dropping of the course 8. The produces a modified schedule without the removed course. | |
| Risk assessment: | High | |
| Importance assessment: | 4/5 | |

5.9.5 System Sequence Diagram

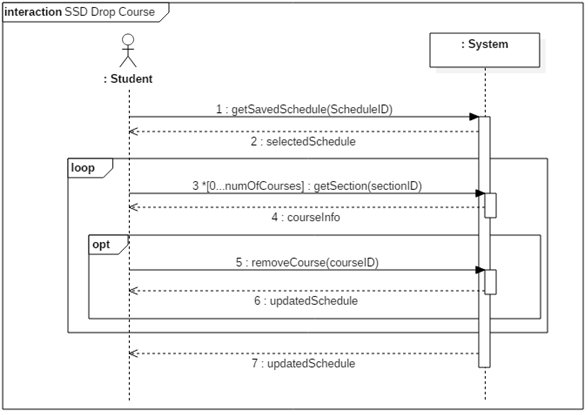


Figure 5.9.1 Drop Course system sequence diagram

5.9.2 Operating Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 9.1 | getSavedSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | getSavedSchedule(scheduleID: long) | |
| Cross-reference: | UC9, UC7, Contract 7.2 | |
| Preconditions: | * User is logged into the system. * User has generated and saved a schedule * User has requested to view a list of saved schedules | |
| Post-conditions: | None | |

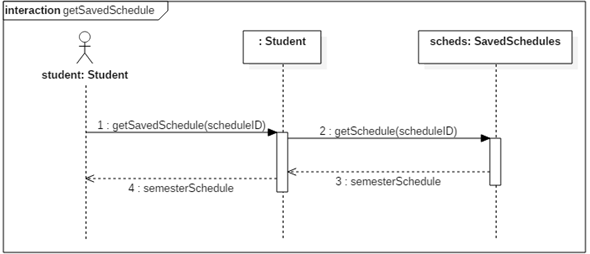


Figure 5.8.2 getSavedSchedule contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 9.2 | getSection | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | viewCourseInfo(sectionID: long) : Section | |
| Cross-reference: | UC9 | |
| Preconditions: | * User is logged into the system. * User has generated and saved a schedule * User has requested to view a saved schedule | |
| Post-conditions: | None | |

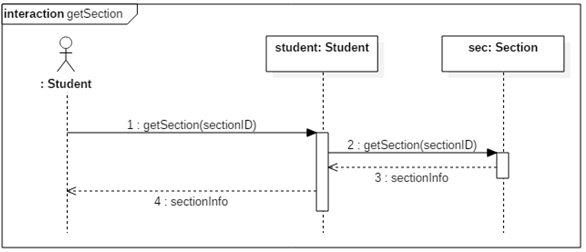
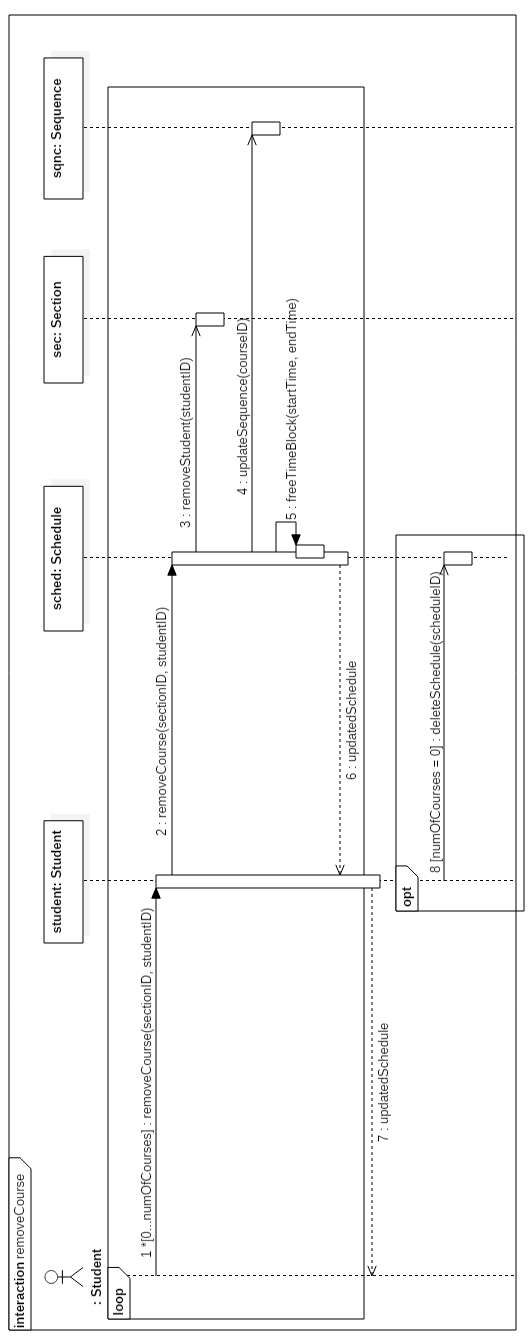


Figure 5.8.3 getSection contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 9.3 | removeCourse | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | removeCourse(sectionID: long, studentID: long) : Schedule | |
| Cross-reference: | UC9 | |
| Preconditions: | * User is logged into the system. * User has generated and saved a schedule * User has requested to view a saved schedule * User has requested to view course registration info | |
| Post-conditions: | * The student will be removed from the section’s registration list (attribute modification, association removed) * The course will be removed from the student’s schedule (association modification) * The student’s course sequence will be updated (attribute modification) | |



5.10 Use Case 10 – Add Course

5.10.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC10 | |
| Use Case Name: | Add Course | |
| Created By: | Lori Dalkin | Last Updated By: Claudia Della Serra |
| Date Created: | February 7, 2016 | Last Revision Date: February 14, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | Add a course to a generated schedule. | |
| Description/Summary: | Specific courses can be added to a schedule generated by a user. The user must search for the course and manually add it to their schedule. | |
| Preconditions: | * The user is logged in to their account * A schedule has been generated | |
| Post-conditions: | The specified course is added to the schedule. | |
| Minimum Guarantee: | The user’s schedule remains unchanged and no new course is added | |
| Basic Flow: | 1. User requests to view schedule. 2. System displays the indicated schedule. 3. User indicates they wish to add a course 4. The system displays a search screen prompting the user to enter the name of the course 5. The user types in the name of the course they wish to add 6. The system displays a list of corresponding courses and sections 7. User selects the specific section of the course they wish to add. 8. The system prompts the user to confirm their selected course 9. The user confirms their wish to add the course 10. The system returns a modified schedule. | |
| Risk assessment: | Medium | |
| Importance assessment: | 3/5 | |

5.10.2 System Sequence Diagram

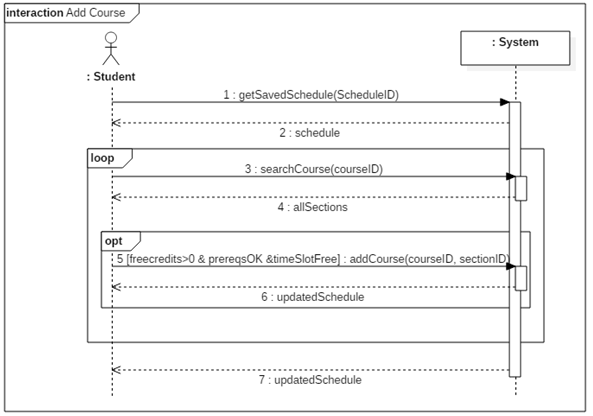


Figure 5.10.1 Add Course system sequence diagram

5.10.3 Operating Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 10.1 | getSavedSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | removeCourse(courseID: long, sectionID: long) | |
| Cross-reference: | UC10, UC7, Contract 7.2 | |
| Preconditions: | * User is logged into the system. * User has generated and saved a schedule * User has requested to view a saved schedule | |
| Post-conditions: | None | |

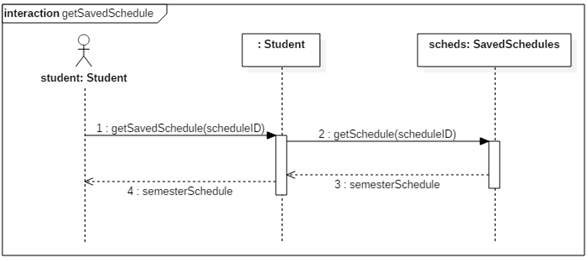


Figure 5.10.2 getSavedSchedule contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 10.2 | searchCourse | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | searchCourse(courseID) : list<Section> | |
| Cross-reference: | UC10 | |
| Preconditions: | * User is logged into the system. * User has generated and saved a schedule * User has requested to view a saved schedule | |
| Post-conditions: | None | |

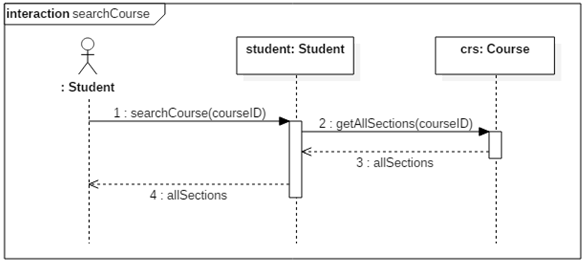


Figure 5.10.3 searchCourse contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 10.3 | addCourse | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | addCourse (courseID, sectionID) | |
| Cross-reference: | UC10 | |
| Preconditions: | * User is logged into the system. * User has generated and saved a schedule * User has requested to view a saved schedule * User has enough free credits to add a class * User has completed the required prerequisites | |
| Post-conditions: | * The student will be added to the section’s registration list (attribute modification, association formed) * The class will be added to the student’s schedule (association formed) | |

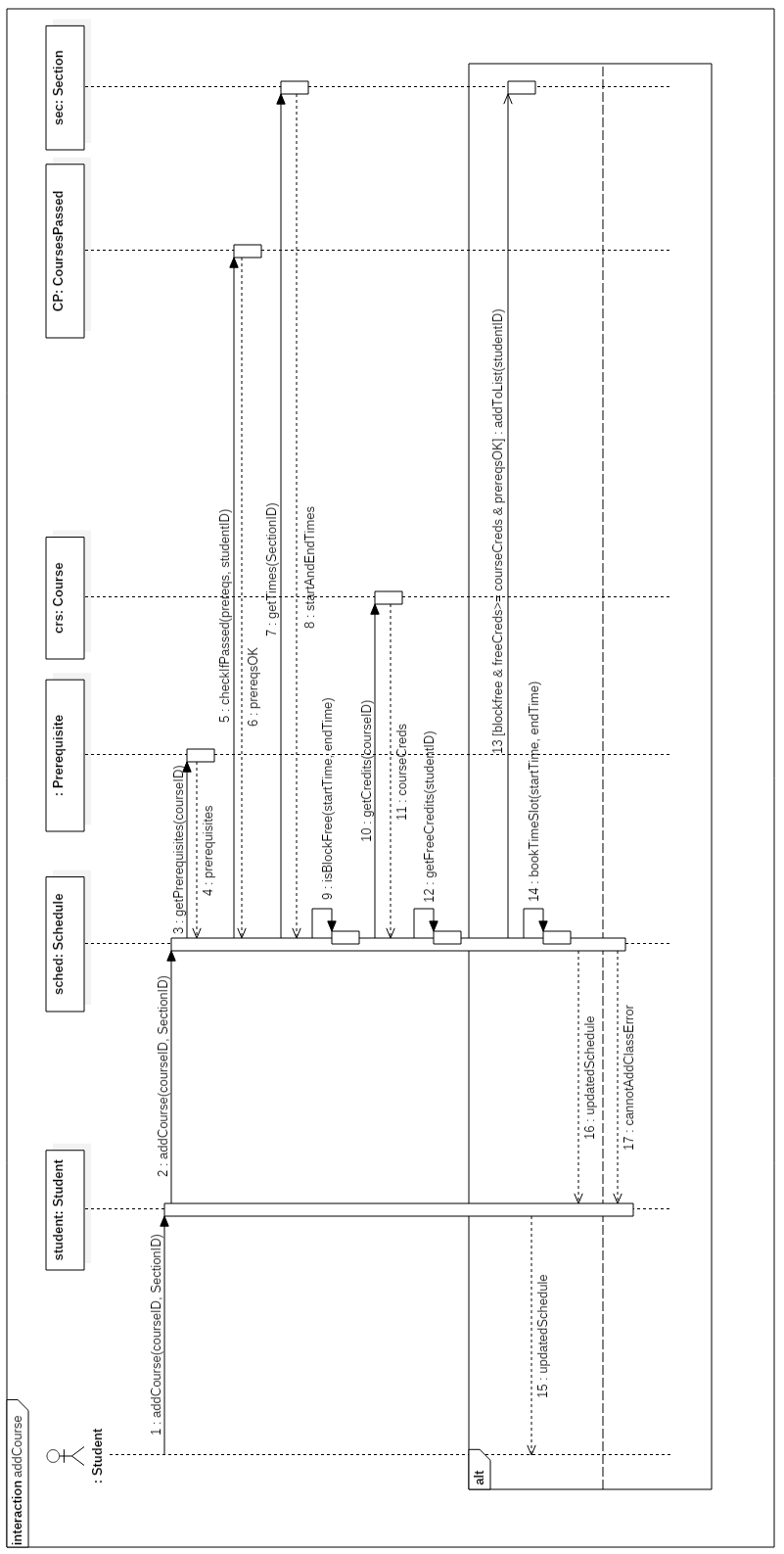


Figure 5.10.4 addCourse contract diagram

5.11 Use Case 11 – Save Generated Schedule

5.11.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC11 | |
| Use Case Name: | Save Generated Schedule | |
| Created By: | Lori Dalkin | Last Updated By: Claudia Della Serra |
| Date Created: | February 7, 2016 | Last Revision Date: February 14, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | The user wishes to save a schedule they have generated. | |
| Description/Summary: | A generated schedule can be saved in order to be accessed and viewed later on by the user. This schedule consists of courses the user is satisfied with and wishes to keep on their schedule. | |
| Preconditions: | * The user is logged on to their account * A schedule has been generated containing at least one course. | |
| Post-conditions: | The system saves the generated schedule to the user’s account. | |
| Minimum Guarantee: | The user’s account remains unchanged. | |
| Basic Flow: | 1. User indicated their wish to generate a schedule 2. The system prompts the user with a list of semesters for which they wish to generate a schedule 3. The user indicates which semester they wish to generate a schedule for 4. The system displays a generated schedule for the indicated semester 5. The user will indicates that they want to save the generated schedule. 6. The system will display a message telling the user that the schedule has been saved. | |
| Risk assessment: | Medium | |
| Importance: | 4/5 | |

5.11.2 System Sequence Diagram

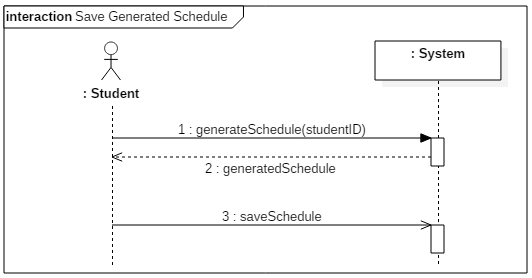


Figure 5.11.1 Save Generated Schedule system sequence diagram

5.11.3 Operating Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 11.1 | generateSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | generateSchedule(studentID) | |
| Cross-reference: | UC11, UC6, contract 6.2 | |
| Preconditions: | * User is logged into the system. * User is enrolled in a program | |
| Postconditions: | None | |

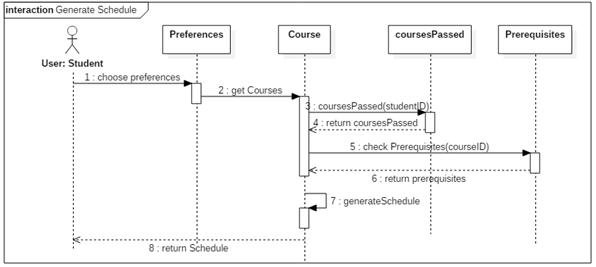


Figure 5.11.2 generateSchedule contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 11.2 | saveGeneratedSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | saveGeneratedSchedule(scheduleID): list<Schedule> | |
| Cross-reference: | UC11 | |
| Preconditions: | * User is logged into the system. * User is enrolled in a program * User has generated a schedule | |
| Post-conditions: | * A schedule for that semester will be saved to the student’s account (association formed) * The schedule will be added to the list of saved schedules (attribute modification) | |

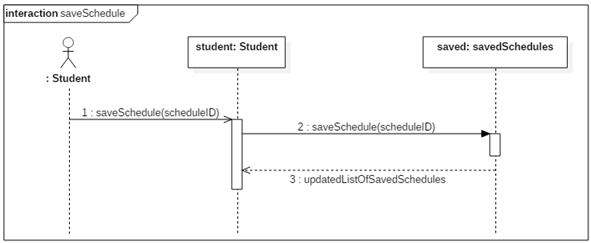


Figure 5.11.3 saveSchedule contract diagram

5.12 Use Case 12 – View Weekly Schedule

5.12.1 Fully Dressed Use Case

|  |  |  |
| --- | --- | --- |
| Use Case ID: | UC12 | |
| Use Case Name: | View Weekly Schedule | |
| Created By: | Bryce Drewery Schoeler | Last Updated By: Claudia Della Serra |
| Date Created: | February 7, 2016 | Last Revision Date: February 14, 2016 |
| Actor(s): | Student | |
| Goal/Actor Goals: | A student wants to view weekly schedule. | |
| Description/Summary: | The student wishes to see their class schedule for the current week. The student will specify a week to be displayed. The system must display the schedule for that week. | |
| Preconditions: | * The student has been authenticated * A schedule has been generated and saved | |
| Post-conditions: | The schedule is shown on screen. | |
| Minimum Guarantee: | The schedule fails to be displayed. | |
| Basic Flow: | 1. Student requests to see a saved schedule. 2. The system prompts the student to select a week. 3. Student selects a week 4. The systems display the schedule for that week | |
| Risk assessment: | Medium | |
| Importance assessment: | 3/5 | |

5.12.2 System Sequence Diagram

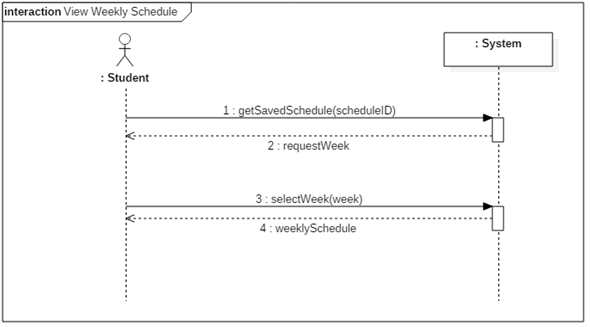


Figure 5.12.1 View Weekly Schedule system sequence diagram

5.12.3 Operating Contract Diagrams

|  |  |  |
| --- | --- | --- |
| Contract 12.1 | getSavedSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | getSavedSchedule(scheduleID) | |
| Cross-reference: | UC12, UC7, contract 7.2 | |
| Preconditions: | * User is logged into the system. * User is enrolled in a program * User has saved a schedule | |
| Post-conditions: | None | |

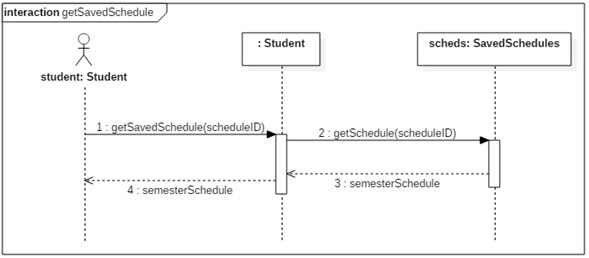


Figure 12.2 getSavedSchedule contract diagram

|  |  |  |
| --- | --- | --- |
| Contract 12.1 | selectWeekSchedule | |
| Created By: | Claudia Della Serra | Last Updated By: Claudia Della Serra |
| Date Created: | March 2, 2016 | Last Revision Date: March 2, 2016 |
| Operation: | selectWeek(date: int, month: int) | |
| Cross-reference: | UC12, | |
| Preconditions: | * User is logged into the system. * User is enrolled in a program * User has saved a schedule | |
| Post-conditions: | None | |

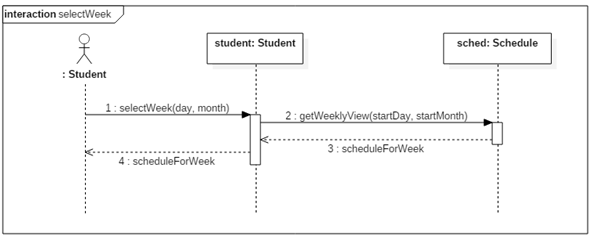


Figure 5.12.3 selectWeek Contract diagram

# Estimation

The current cost and time estimation has been updated from previous estimations made in deliverable 1 by considering the estimated lines of code that will be produced in the implementation of the TimeTurner system. An estimated person-months amount was calculated, and then this amount applied to previous estimations made in conjunction with estimations based on time-logs taken from activities performed in the production of the Deliverable 2 document.

6.1 Function Point Estimation

|  |  |  |
| --- | --- | --- |
| **Internal Logical Files** | | |
| **Files** | **DETs** | **RETs** |
| **Completed Courses** | UserID | Courses grouped by User |
|  | CourseID |  |
| **Course** | Course\_code | Courses grouped by Course Code (example: COMP, SOEN, etc) |
|  | Course\_description |  |
|  | cType |  |
|  | Credits |  |
| **Prerequisite** | courseID (repeat) | Courses grouped by the course they are the prerequisite for |
|  | prerequisiteID |  |
| **Section** | courseID (repeat) |  |
|  | Kind |  |
|  | Sections |  |
|  | Days |  |
|  | Start\_time |  |
|  | End\_time |  |
|  | semester |  |
| **Subsection** | sectionID |  |
|  | courseID (repeat) |  |
|  | Kind |  |
|  | Sections (repeat) |  |
|  | Days (repeat) |  |
|  | Start\_time (repeat) |  |
|  | End\_time (repeat) |  |
|  | Semester (repeat) |  |
| **User** | Username |  |
|  | Firstname |  |
|  | Lastname |  |
|  | netName |  |
|  | Password |  |
| **User\_schedule** | ID | Schedules grouped by the user they belong to |
|  | userID (repeat) |  |
|  | sectionID(repeat) |  |
|  | Year |  |
| **Total:** | 22 | 4 |

|  |  |  |
| --- | --- | --- |
| **External Inputs** | | |
| **Method** | **DETs** | **FTRs** |
| **saveSchedule** | scheduleID | User\_schedule |
| **addCourse** | courseID | User\_schedule (repeat) |
|  | sectionID | Section |
|  |  | Subsection |
| **deleteSchedule** | scheduleID (repeat) | User\_schedule (repeat) |
| **freeTimeBlock** | startTime |  |
|  | endTime | User\_schedule (repeat) |
| **removeCourse** | sectionID (repeat) | User\_schedule (repeat) |
|  | studentID | Section (repeat) |
|  |  | Subsection (repeat) |
| **updateSequence** | courseID (repeat) |  |
| **addToList** | studentID (repeat) | Section (repeat) |
| **removeStudent** | studentiD (repeat) | Section (repeat) |
| **Login** | adminID | User |
|  | studentID (repeat) |  |
|  | password |  |
| **Logout** | adminID (repeat) | User (repeat) |
|  | studentID (repeat) |  |
| **Total** | 8 | 4 |

|  |  |  |
| --- | --- | --- |
| **External Outputs** | | |
| **Method** | **DETs** | **FTRs** |
| **getFreeCredits** | studentID | User\_schedule |
| **isBlockFree** | startTime | User\_Schedule (repeat) |
|  | endTime |  |
| **removeCourse** | SectionID | User\_schedule (repeat) |
|  | studentID (repeat) | Section |
|  |  | Subsection |
| **checkPassed** | prereqs | Courses\_complete |
| **createCourseSequence** | StudentID (repeat) | Course |
|  |  | Section (repeat) |
| **getCourseList** |  | Course (repeat) |
| **getCourseInfo** |  | Course (repeat) |
|  |  | Section (repeat) |
|  |  | Prerequisite |
| **generateSchedule** | studentID (repeat) | Course(repeat) |
|  |  | Section (repeat) |
|  |  | Subsection (repeat) |
|  |  | Prerequisite (repeat) |
| **Total** | 5 | 6 |

|  |  |  |
| --- | --- | --- |
| **External Queries** | | |
| **Method** | **DETs** | **FTRs** |
| **getSavedSchedule** | scheduleID | User\_schedule |
| **searchCourse** | courseID | Course |
|  |  | Section |
|  |  | Subsection |
| **selectWeek** | Day | User\_schedule (repeat) |
|  | Month |  |
| **viewAcademicRecord** | studentID | Courses\_complete |
| **bookTimeSlot** | startTime | User\_schedule (repeat) |
|  | endTime |  |
| **getSchedule** | scheduleID (repeat) | User\_schedule (repeat) |
| **getWeeklyView** | Day (repeat) | User\_schedule (repeat) |
|  | Month (repeat) |  |
| **gatherCoursesPassed** | studentID (repeat) | Courses\_complete (repeat) |
| **getAllSections** | courseID (repeat) | Course (repeat) |
|  |  | Section (repeat) |
| **getCredits** | courseID (repeat) | Course (repeat) |
| **getSection** | sectionID (repeat) | Section (repeat) |
| **getTimes** | sectionID (repeat) | Section (repeat) |
| **viewCourseSequence** | studentID(repeat) | User\_schedule (repeat) |
| **Total** | 7 | 5 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Function Type** | | **Functional Complexity** | | | | | |  | | **Complexity Totals** | | | | | | **Function Type Totals** | |
| ILF | | 0 Low | | | | | | X 7 | | 0 | | | | | |  | |
|  | | 10 Average | | | | | | X10 | | 100 | | | | | |  | |
|  | | 0 High | | | | | | X15 | | 0 | | | | | | 100 | |
|  | | | | | | | | | | | | | | | | | |
| EIF | | | | 0 Low | | | X 5 | | 0 | | | | | |  | | |
|  | | | | 0 Average | | | X7 | | 0 | | | | | |  | | |
|  | | | | 0 High | | | X10 | | 0 | | | | | | 0 | | |
|  | | | | | | | | | | | | | | | | | |
| EI | | | | | 0 Low | | | X3 | | | 0 | | | | |  | |
|  | | | | | 4 Average | | | X4 | | | 16 | | | | |  | |
|  | | | | | 0 High | | | X6 | | | 0 | | | | | 16 | |
|  | | | | | | | | | | | | | | | | | |
| EQ | | | 0 Low | | | | X 3 | | | | | | 0 |  | | | |
|  | | | 0 Average | | | | X4 | | | | | | 0 |  | | | |
|  | | | 6 High | | | | X6 | | | | | | 36 | 36 | | | |
|  | | | | | | | | | | | | | | | | | |
| EO | 0 Low | | | | | X 4 | | | | | | 0 | |  | | | |
|  | 5 Average | | | | | X5 | | | | | | 25 | |  | | | |
|  | 0 High | | | | | X7 | | | | | | 0 | | 25 | | | |
|  | | | | | | | | | | | | | | | | | |
| **Unadjusted Function Point Count** | | | | | | | | | | | | | | | | | 177 |
| **SLOC/FP for PHP [1]** | | | | | | | | | | | | | | | | | 67 |
| **Total SLOC** | | | | | | | | | | | | | | | | | 11859 |
| **COCOMO II estimated person-months effort** | | | | | | | | | | | | | | | | | 43.8 |
| **Months of effort per person (team of 12)** | | | | | | | | | | | | | | | | | 3.65 |

[1] taken from <https://www.cs.helsinki.fi/u/taina/ohtu/fp.html>

Given the above estimated lines of code, this value was put into the COCOMO II engine, with the Software Scale Drivers modified as follows:

* Pecedentedness: Low
  + Due to the fact that not very much experience in programming a large system such as this one has been previously incurred by many of the team members.
* Architecture/Risk Resolution: high
  + Due to the fact that much planning in architecture and design has been undergone, and that risk assessment and analysis has been performed since the start of the project.
* Process Maturity: low
  + Due to the fact that the team has only begun using software process management, and does not have much experience in handling projects in this manner
* Development flexibility: high
  + Due to the fact that very vague requirements were given by instructors, thus the flexibility to implement these requirements, and create our own, was very high.
* Team cohesion: high
  + The team has had much experience working together on previous assignments, labs, etc. thus team cohesion is high and group work and communication are excellent.

6. 2 Module Estimations

Given the estimated time of 43.8 person-months, and previous time estimates that were performed in the first version of the project, new time estimates have been made. Although usual person-month time estimates are roughly 160 hours/person, given that all team members are currently full-time students and working on other projects throughout the course of this project development, our person-months are based on a rough estimate of 4.85 hours per person per task; such an estimate is taken from the time logs of one team member working on the Deliverable 2 task of creating system sequence and contract diagrams (times were divided by 2 due to the fact that all system sequence diagrams were implemented rather than just 2 major ones). This results in a 424.86 person-hours estimate for the entire project. Thus, previous estimates are updated as follows:

6.2.1 Deliverable 0 Estimates

|  |  |  |
| --- | --- | --- |
| **Task Names** | **Cost (Hours)** | **Revised Cost (hours)** |
| Domain Model | 6 | 6 |
| Identification of Technologies | 12 | 12 |
| Team Members | 0.5 | 0.5 |
| Implementation Discussion | 5 | 5 |
| DMO Descriptions | 6 | 6 |
| **Total** | **29.5** | **29.5** |

6.2.2 Deliverable 1 Estimates

|  |  |  |
| --- | --- | --- |
| **Task Names** | **Cost (Hours)** | **Revised Cost (Hours)** |
| Requirements | 10 | 10 |
| Use Case Diagrams/Models | 8 | 8 |
| Resources | 5 | 5 |
| Scope | 5 | 5 |
| Architecture | 7 | 7 |
| **Total** | **35** | **35** |

6.2.3 Deliverable 2 Estimates

|  |  |  |
| --- | --- | --- |
| **Task Names** | **Cost (Hours)** | **Revised Cost (Hours)** |
| Architecture Design/Revision | 100 | 29.1 |
| Use Case Implementation | 50 | 38.8 |
| Design Discussion/Implementation | 25 | 116.4 |
| Prototyping | 15 | 9.7 |
| **Total** | **190** | **194** |

6.2.4 Deliverable 3 Estimates

|  |  |  |
| --- | --- | --- |
| **Task Names** | **Cost (Hours)** | **Revised Cost (Hours)** |
| Unit/Stress/Security Testing | 30 | 29.1 |
| Debugging/Optimizing code based on test cases | 15 | 19.4 |
| User Performance Test | 20 | 29.1 |
| Debugging/Optimizing code based on user tests | 10 | 19.4 |
| **Total** | **75** | **97** |

6.3.5 Deliverable 4 Estimates

|  |  |  |
| --- | --- | --- |
| **Task Names** | **Cost (Hours)** | **Revised Cost (Hours)** |
| Final Optimizations | 20 | 29.1 |
| Compiling data into one file | 30 | 9.7 |
| Final Test on System | 10 | 9.7 |
| **Total** | **60** | **48.5** |

6.3.4 Total Deliverable Estimates

|  |  |  |
| --- | --- | --- |
| **Deliverable Names** | **Cost (Hours)** | **Revised Cost (Hours)** |
| Deliverable 0 | 29.5 | 29.5 |
| Deliverable 1 | 35 | 35 |
| Deliverable 2 | 190 | 194 |
| Deliverable 3 | 75 | 97 |
| Deliverable 4 | 60 | 48.5 |
| **Total** | **389.5** | **404** |

# Rapid Prototyping and Risk